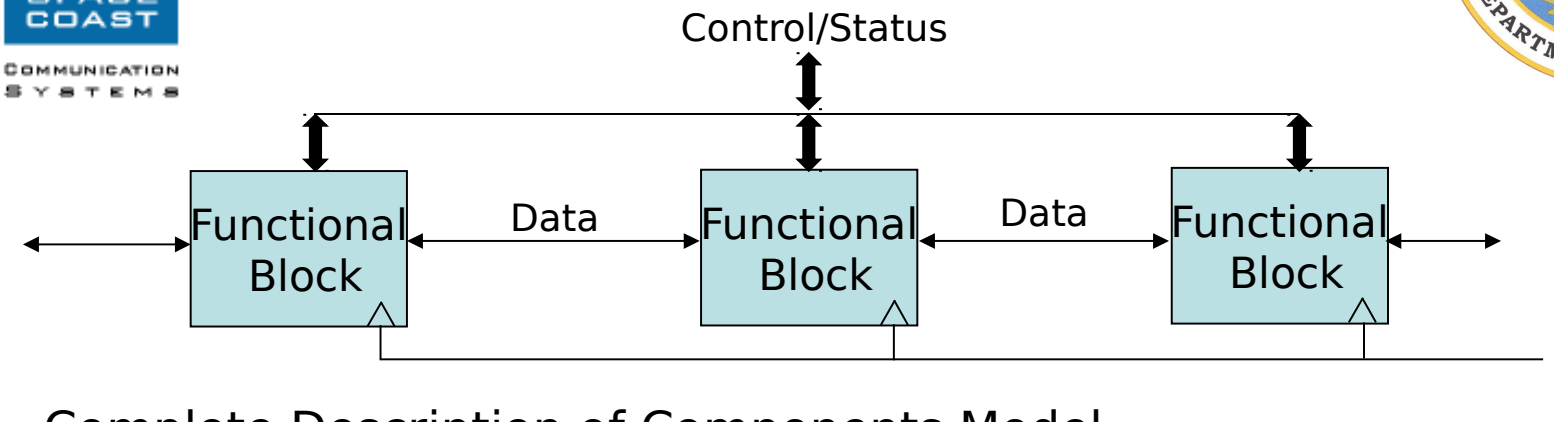
Components Concept

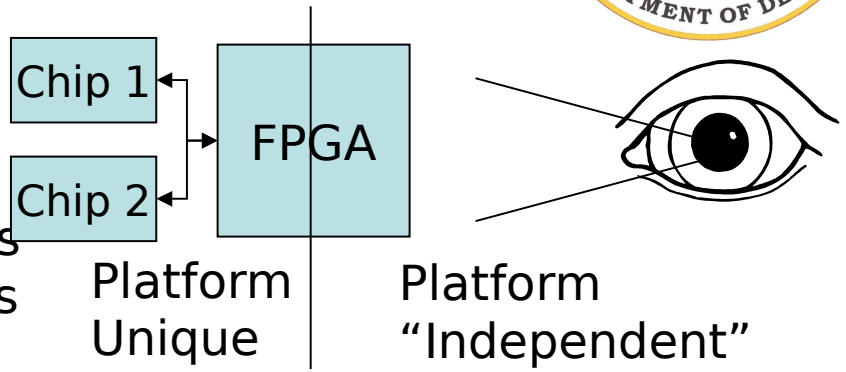


Complete Description of Components Model

1. Placement and Implementation of Functional Block
   - might be ASIC, FPGA, DSP, GPP
   - portability will always involve cost ➔ how to minimize?
2. Description and Connectivity of Data Path
   - multi-layered definition from pinout to ACK/NAK to IDL
3. Timing and Synchronization Methods
   - current SCA essentially asynchronous
   - logical source of clock foreground = A/D-D/A sample clock
4. Control and Status API's

Implementation Space

Domains: DSP ➔ with RTOS
DSP ➔ without RTOS
FPGA → SPS functional blocks
FPGA ➔ glue logic to chipsets



Primary benefit of SCA ➔ hides the implementation (abstraction)
Current SCA: - All implementations wrapped in CORBA as required to make the resource visible to the domain
- Deployments not required to use IDL if performance dictate
SCA extension: - hide the implementation (abstraction)
- allow commonalized access at layers underneath CORBA

IMPORTANT:
As much as SCA is about architecture, CORBA (mandated) is about implemen
Corollary: there is nothing wrong in mandating COTS implementation
Challenge: COTS mandates want to be supportable and extensible to 201

Device Definitions

Radio Device vs. Waveform Device

An object associated with the radio that is instantiated at radio power on. Available as an allocatable resource to any waveform.

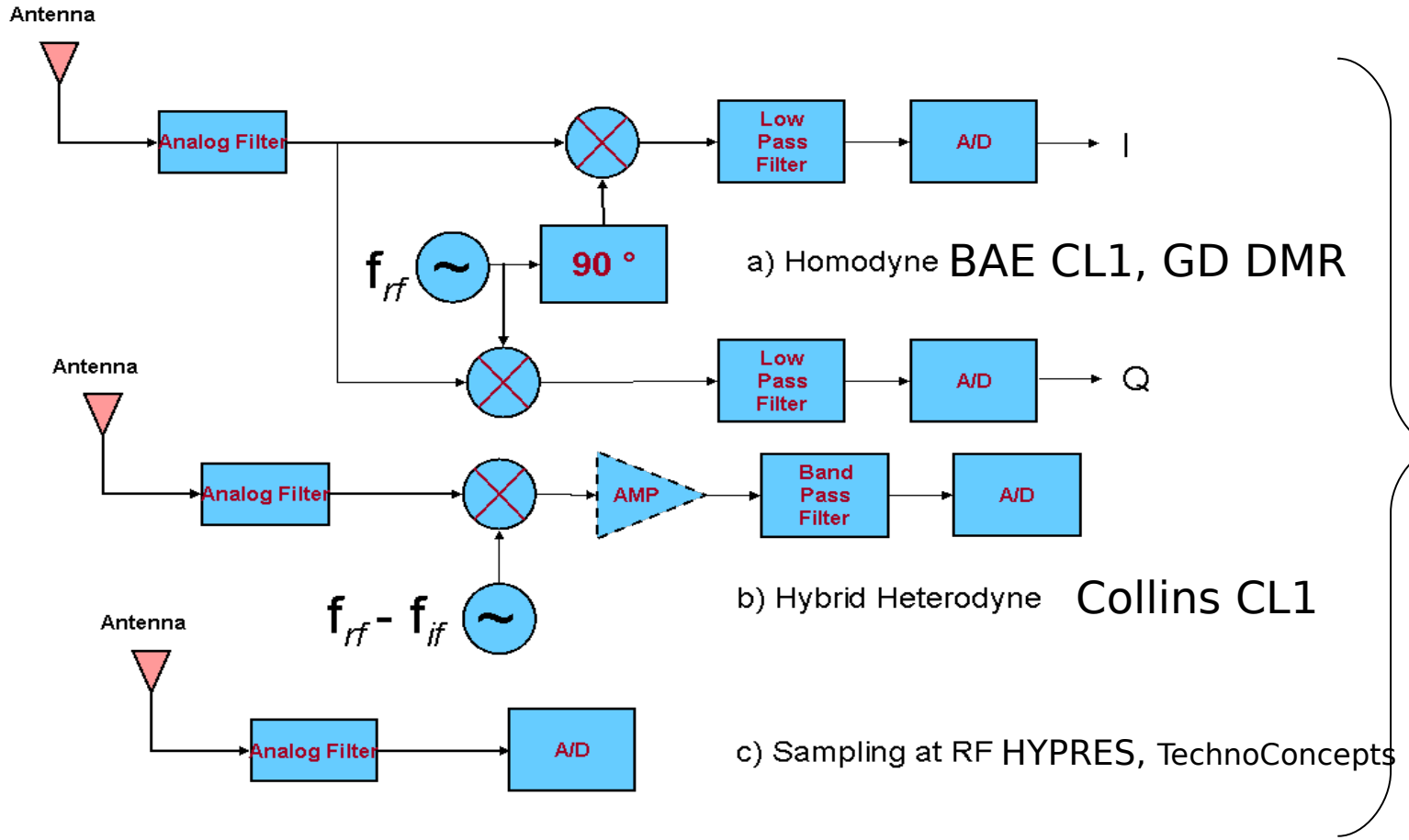
An object associated with an application that is executed/loaded after the radio is booted. Multiply instantiable if JTR set supports

Service Device, An object optionally associated with the radio available as an allocatable resource to all applications

JTRS State of Devices

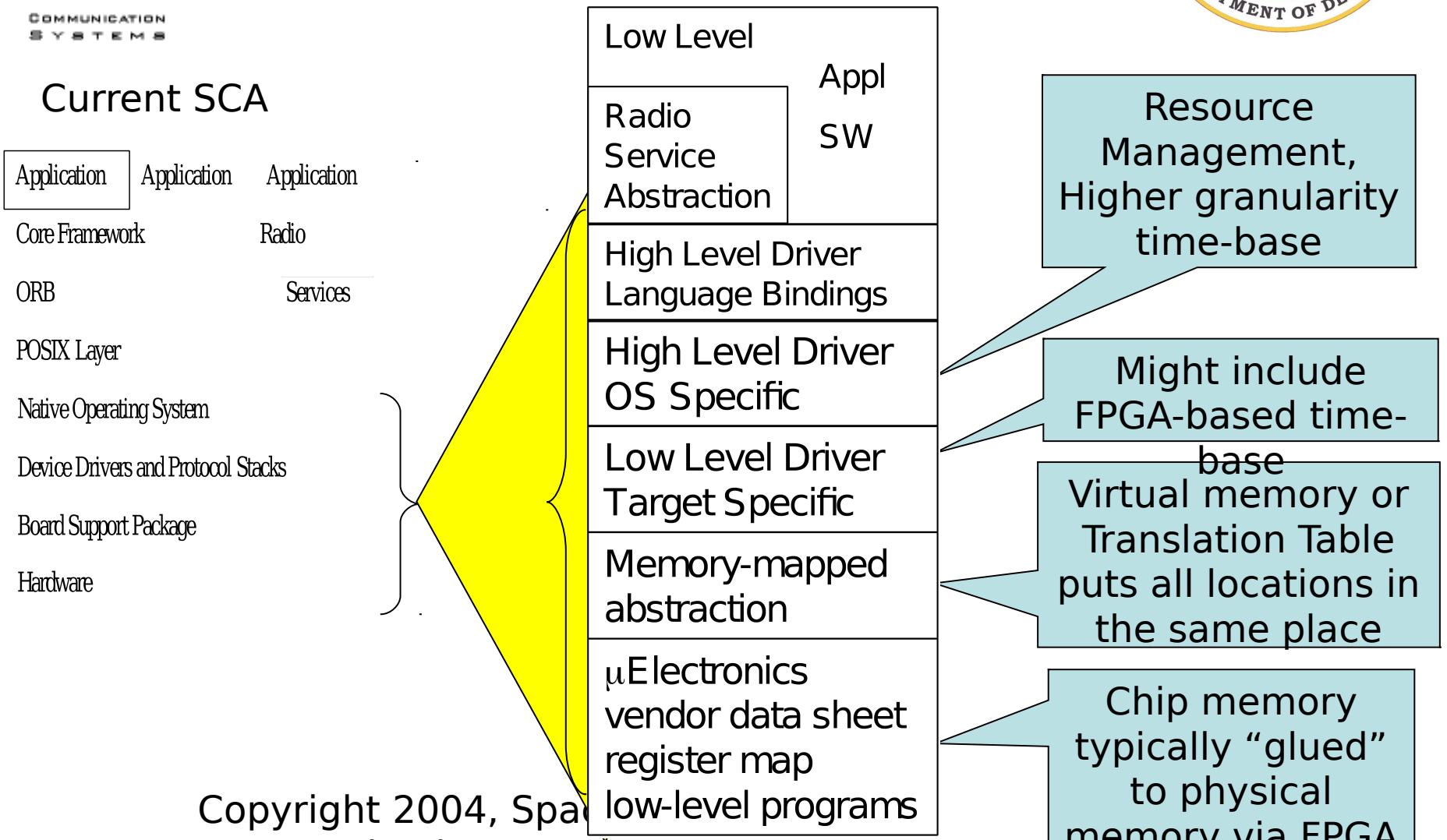
Radio Devices – none published
Waveform Devices – none published
Service Devices – none published

Front End Configurations



SCA Extension must accommodate all configurations

Multiple Layers of Abstraction



Simple API Example

The following API set alludes to the architectural need to be able link the timing subsystem into the command path

- Forget hardware precision constraints – use engineering units

  setFrequency(double frequencyInHertz)

Implementation 1

- Includes no notion of time – when to strobe the frequency?

  setFrequency(double frequencyInHertz, double timeInMsec)

- Definitions of time? how is it calibrated? (see Time Reference API)

Implementation 2

- What if hop-times are smaller than the OS tic timer?

setFrequency(double *frequencyInHertz, double *timeInMsec, int arraySize);

Implementation 3